(From Mike Iatauro)

I finished implementing the second search operator and the related heuristic yesterday evening, so here's a brief overview of how to use the new stuff:

Instant threats (just add water!) are managed by the SAVHThreatManager, so your Solver config needs an "<SAVHThreatManager>" element, at the same level as other flaw managers. The SAVHThreatManager has one configuration attribute, "order", which is a comma-delimited string of order keywords. These are evaluated starting on the left, and moving to the right to break ties between flawed instants. The currently available orders are:

* most/least to order by most or least flawed
* earliest/latest to order by earliest or latest time
* upper/lower to order by upper or lower level flaws

For example, if you wanted to choose flaws that are latest, most flawed, and prefer upper level over lower level flaws, the SAVHThreatManager opening tag would look like this:

```
<SAVHThreatManager order="latest,most,upper">
```

The default if no order is given is "lower,most,earliest".

There is only one type of flaw handler for Instant, so all FlawHandler elements under the SAVHThreatManager should have their "component" attribute set to "SAVHThreatHandler".
The only standard FlawHandler attribute that is supported is "class-match".
In addition, there are four configuration attributes: order, filter, constraint, and iterate.

The order attribute is exactly like the SAVHThreatManager order attribute, but it accepts the following orders (all orders that end in "Predecessor" have a counterpart that ends in "Successor"):

* leastImpact will order choices by least estimated temporal impact
* earliestPredecessor/latestPredecessor will order based on the earliest lower bound or latest upper bound of the predecessor's time
* longestPredecessor/shortestPredecessor will order based on the distance between the lower and upper bound of the predecessor's time
* ascendingKeyPredecessor/descendingKeyPredecessor will order based on the entity key of the predecessor's time

The default is "ascendingKeyPredecessor,ascendingKeySuccessor", which is also tacked on to the end of the order string if the order options you provide don't form a total order.

The filter attribute takes one of "none", which filters no choices, "predecessorNot", which will filter in choices where the predecessor is not contributing to the appropriate level, "successor", which filters in choices where the successor is contributing to the appropriate level, and "both", which does... both. The default is "none".

The constraint attribute controls the available search operators and their order. It takes one of "predecesOnly", "precedesFirst", "concurrentOnly", and "concurrentFirst", which do what you'd expect. The default is "precedesOnly".

Finally, the iterate attribute controls whether the decision point exhausts the set of pairs before moving on to the next operator or vice-versa. It takes one of "pairFirst" or "constraintFirst". Obviously this is only meaningful if the constraint attribute is "precedesFirst" or "concurrentFirst".

An example follows:

```
<!-- This manager prefers the most flawed, earliest, lower level flaws -->
<SAVHThreatManager defaultPriority="0" order="lower,most,earliest">
  <!-- Decisions on the Foo class will filter their choices by successor contributing, only create prec
       least estimated temporal impact and the lower bound of the predecessor's timepoint  -->
  <FlawHandler class-match="Foo" filter="successor" constraint="precedesOnly" order="leastImpact,earlie

  <!-- Everything else will use both filters, create "concurrent" constraints before "precedes" constra
       before moving on to the next pair, and order by the longest distance between the bounds of the s
       between the bounds of the predecessor's time -->
  <FlawHandler filter="both" constraint="concurrentFirst" iterate="constraintFirst" order="longestSucce
</SAVHThreatManager>
```